plotting_pandas

November 30, 2022

```
In [1]: import pandas as pd

        % matplotlib inline
```

Walking through some of the most common plots that can be created with pandas. Pandas is nice for quick insights, but you will need to use matplotlib to really dive into details and customize your visualizations.

```
In [ ]:
```

```
In [2]: df = pd.read_csv("census_income_data.csv")
        df.head().T
```

```
Out[2]:                          0                  1                  2  \
        age                     39                 50                 38
        workclass        State-gov     Self-emp-not-inc            Private
        fnlwgt               77516              83311             215646
        education        Bachelors          Bachelors            HS-grad
        education-num           13                 13                  9
        marital-status   Never-married  Married-civ-spouse           Divorced
        occupation        Adm-clerical    Exec-managerial  Handlers-cleaners
        relationship     Not-in-family            Husband      Not-in-family
        race                 White              White              White
        sex                   Male               Male               Male
        capital-gain          2174                  0                  0
        capital-loss             0                  0                  0
        hours-per-week          40                 13                 40
        native-country   United-States      United-States      United-States
        income                <=50K              <=50K              <=50K

                                 3                  4
        age                     53                 28
        workclass          Private            Private
        fnlwgt              234721             338409
        education             11th          Bachelors
        education-num            7                 13
        marital-status   Married-civ-spouse  Married-civ-spouse
        occupation     Handlers-cleaners    Prof-specialty
```

```
relationship                Husband                 Wife
race                          Black                 Black
sex                            Male                Female
capital-gain                      0                     0
capital-loss                      0                     0
hours-per-week                   40                    40
native-country        United-States                 Cuba
income                        <=50K                 <=50K
```

```
In [3]: df.info()
```

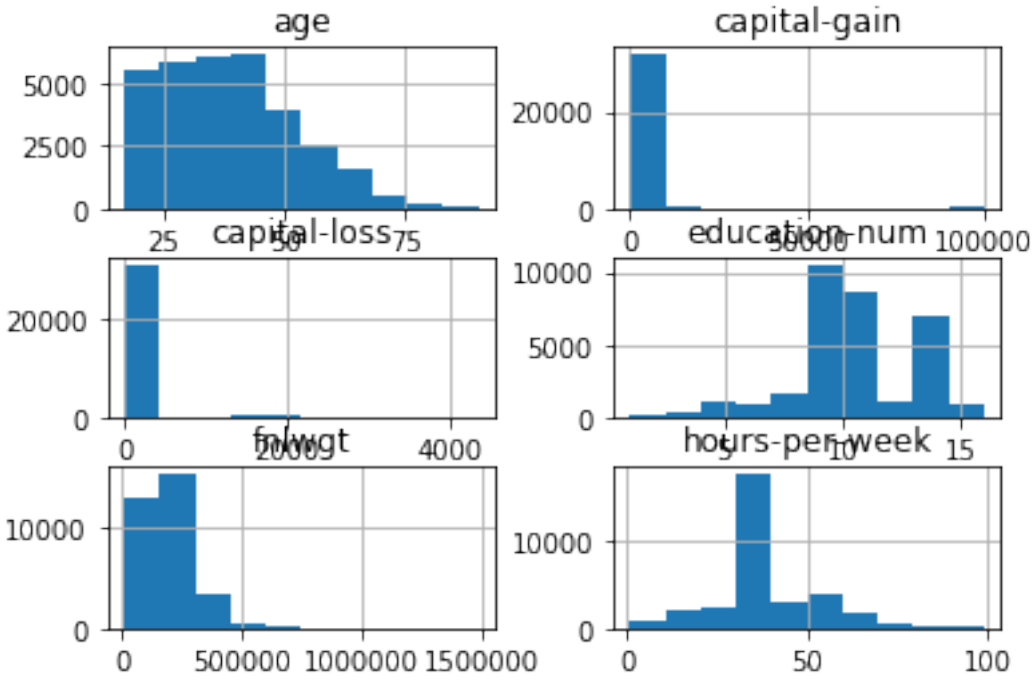
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
age               32561 non-null int64
workclass         30725 non-null object
fnlwgt            32561 non-null int64
education         32561 non-null object
education-num     32561 non-null int64
marital-status    32561 non-null object
occupation        30718 non-null object
relationship      32561 non-null object
race              32561 non-null object
sex               32561 non-null object
capital-gain      32561 non-null int64
capital-loss      32561 non-null int64
hours-per-week    32561 non-null int64
native-country    31978 non-null object
income            32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
In [ ]:
```

A quick ways to view all numercial columns in a dataframe is `.hist()` function, which can be call directly on the dataframe.

```
In [4]: df.hist()
```

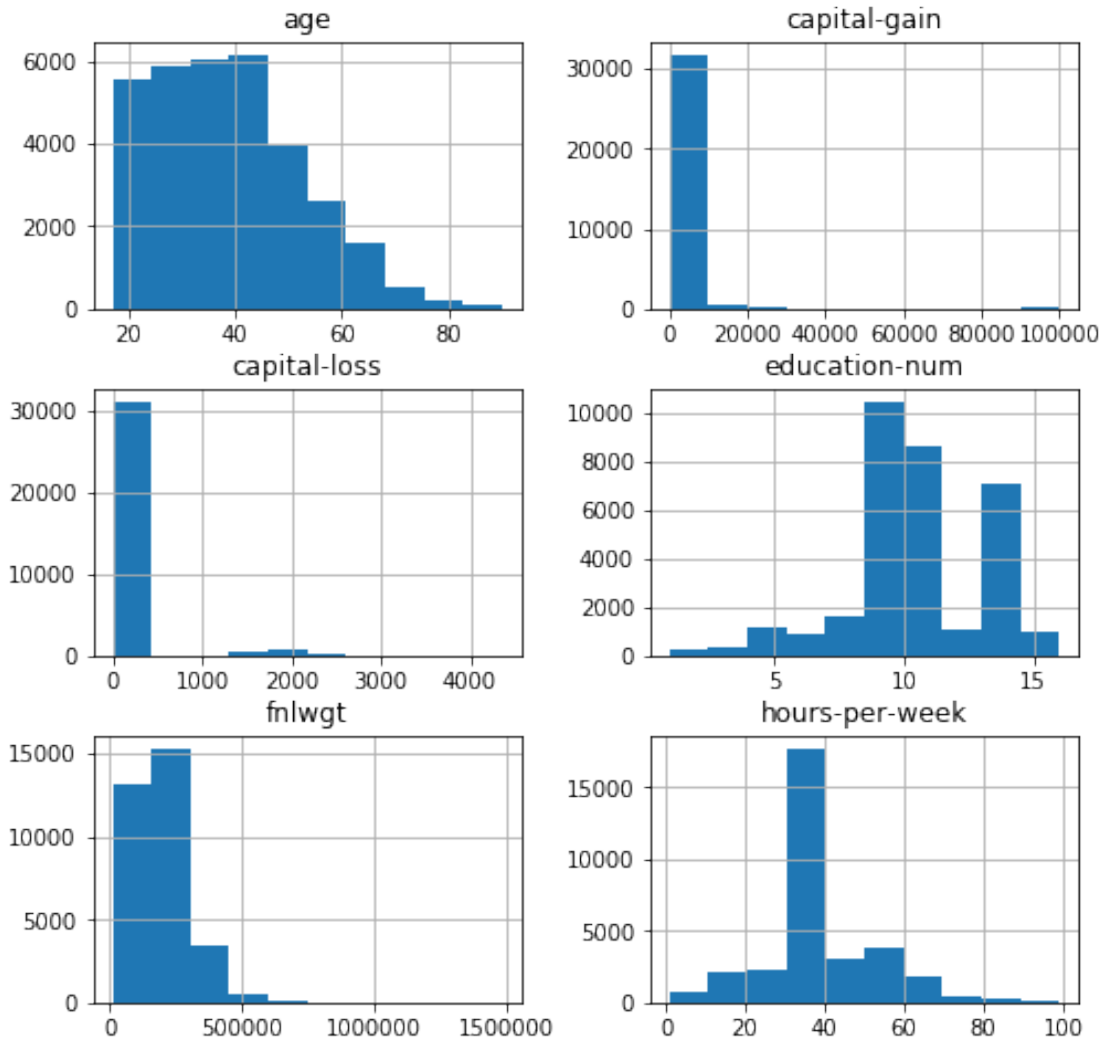
```
Out[4]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fad2108c940>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7fad1effb7b8>],
               [<matplotlib.axes._subplots.AxesSubplot object at 0x7fad1efb9a58>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7fad1ef76a58>],
               [<matplotlib.axes._subplots.AxesSubplot object at 0x7fad1efa7908>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7fad1efa7940>]], dtype=object
```



The graph are way too crowded. It will be good to make the figures size bigger.

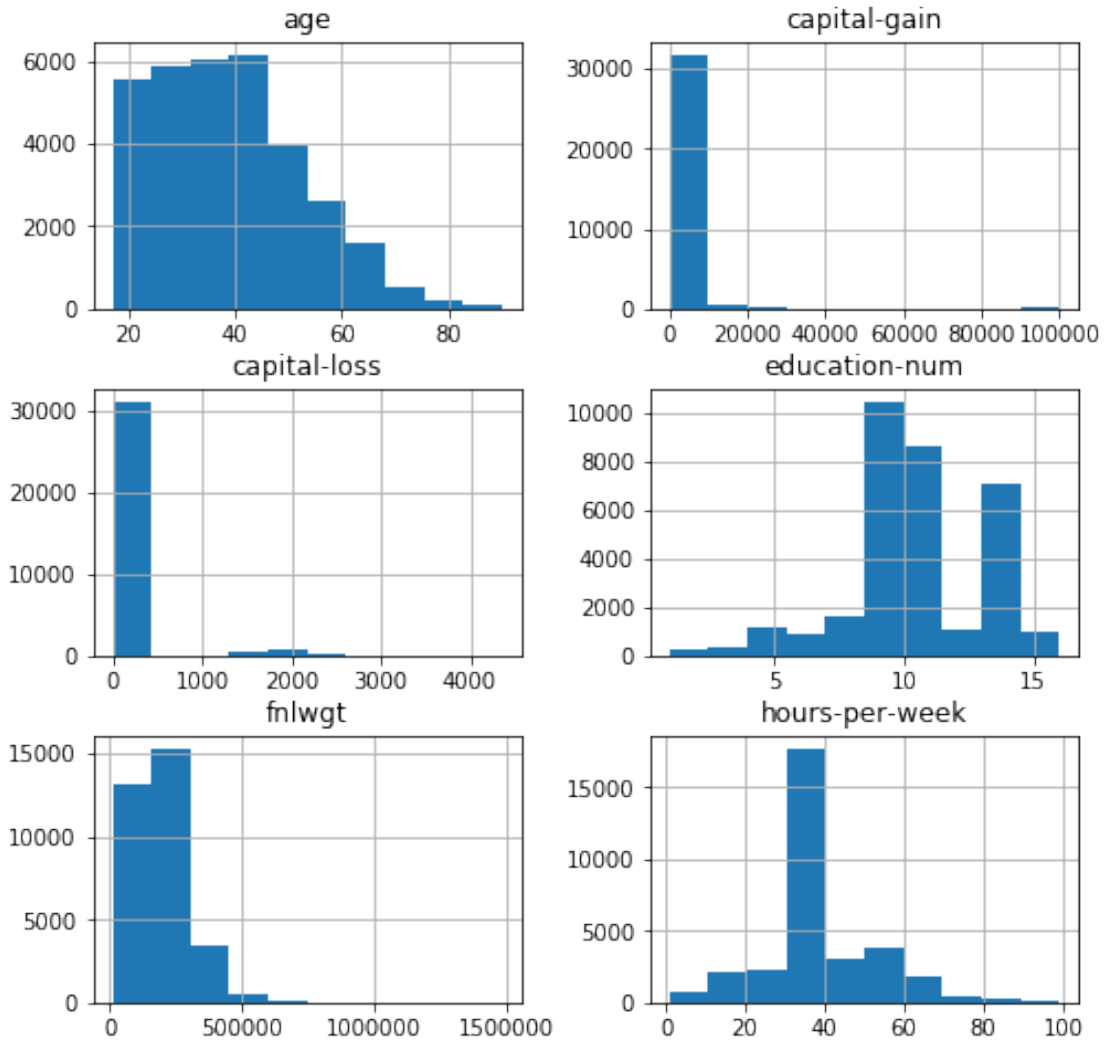
```
In [5]: df.hist(figsize=(8,8))
```

```
Out[5]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fad1eee17f0>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7fad1ed76e10>],
               [<matplotlib.axes._subplots.AxesSubplot object at 0x7fad1edadd30>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7fad1ed6ada0>],
               [<matplotlib.axes._subplots.AxesSubplot object at 0x7fad1ed26da0>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x7fad1ed26dd8>]], dtype=object
```



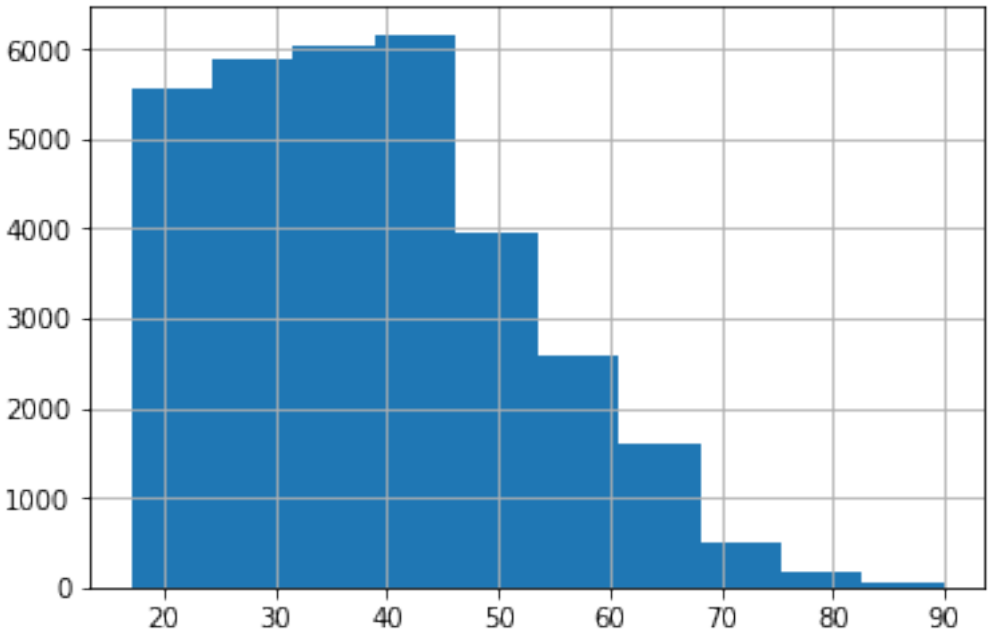
It is much better, but it will be good to suppress those unwanted output.

```
In [6]: # Put a comma to the end of the line, to suppress unwanted outputs
        df.hist(figsize=(8,8));
```



The `.hist()` function can also be called on a pandas series object

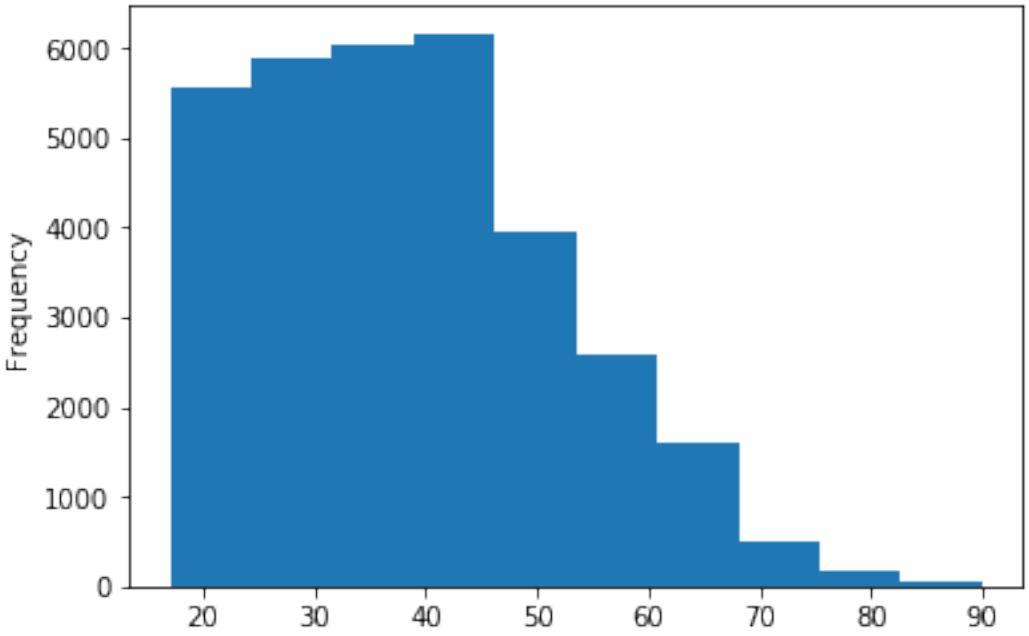
```
In [7]: df["age"].hist();
```



In [ ]:

Instead of using `.hist()`, there is the possibility to use the more general `.plot()` function and specify the type of plot in a parameter

```
In [8]: df["age"].plot(kind="hist");
```



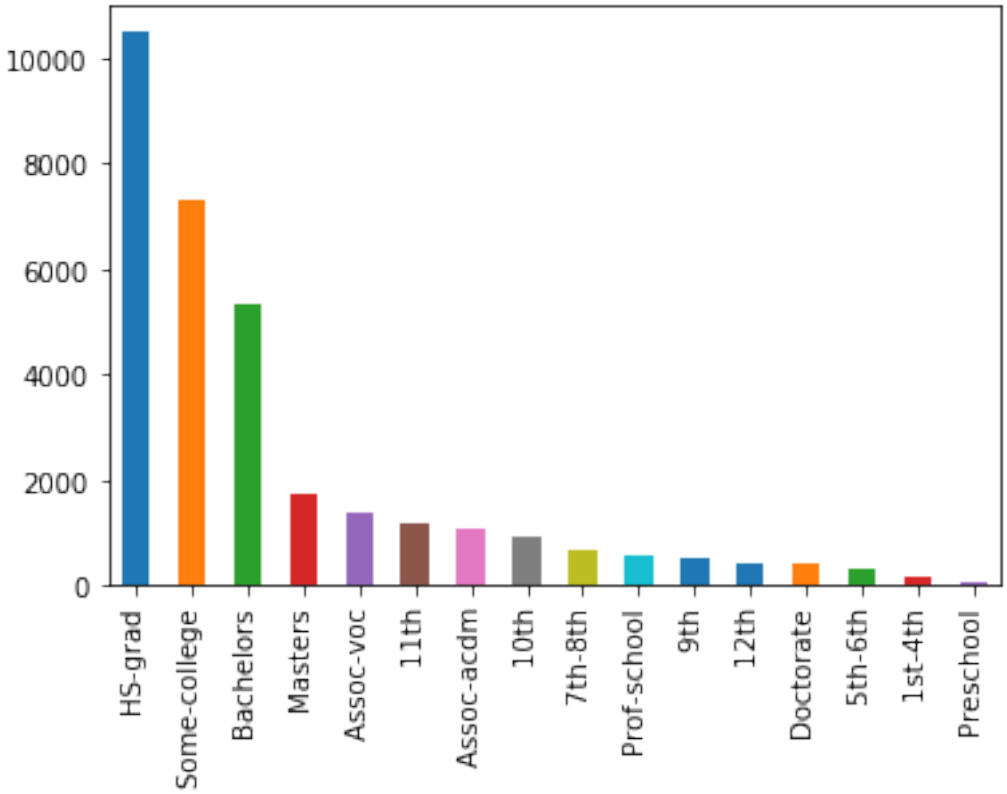
```
In [ ]:
```

Next, let us plot a bar chart for the education column. For that, we need the count for each distinct value

```
In [9]: # Count each distinct value for the education column
        df["education"].value_counts()
```

```
Out[9]: HS-grad         10501
        Some-college     7291
        Bachelors        5355
        Masters          1723
        Assoc-voc        1382
        11th             1175
        Assoc-acdm       1067
        10th              933
        7th-8th           646
        Prof-school       576
        9th               514
        12th              433
        Doctorate         413
        5th-6th           333
        1st-4th           168
        Preschool          51
        Name: education, dtype: int64
```

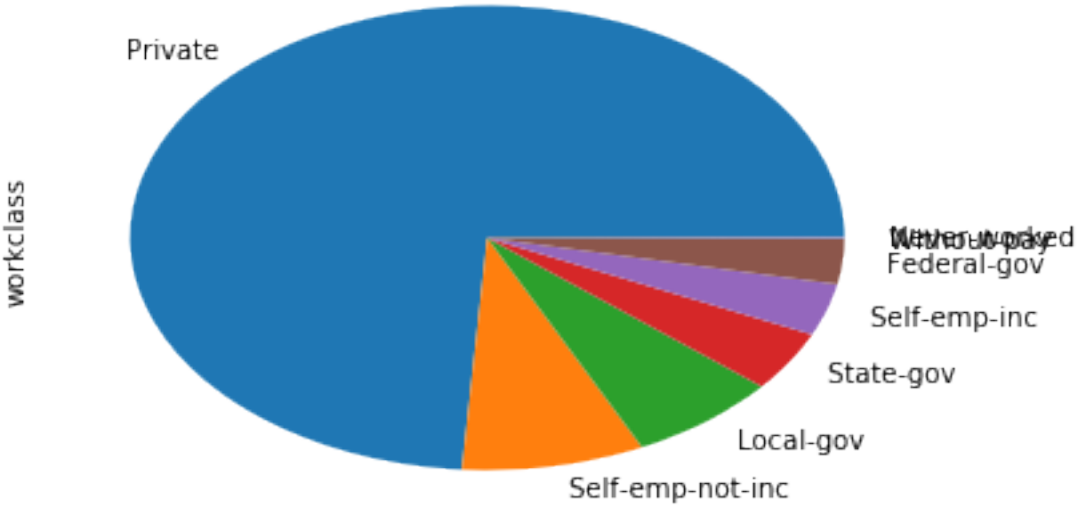
```
In [10]: # Plot a bar chart to illustrate distinct values of the education column
         df["education"].value_counts().plot(kind="bar");
```



In [ ]:

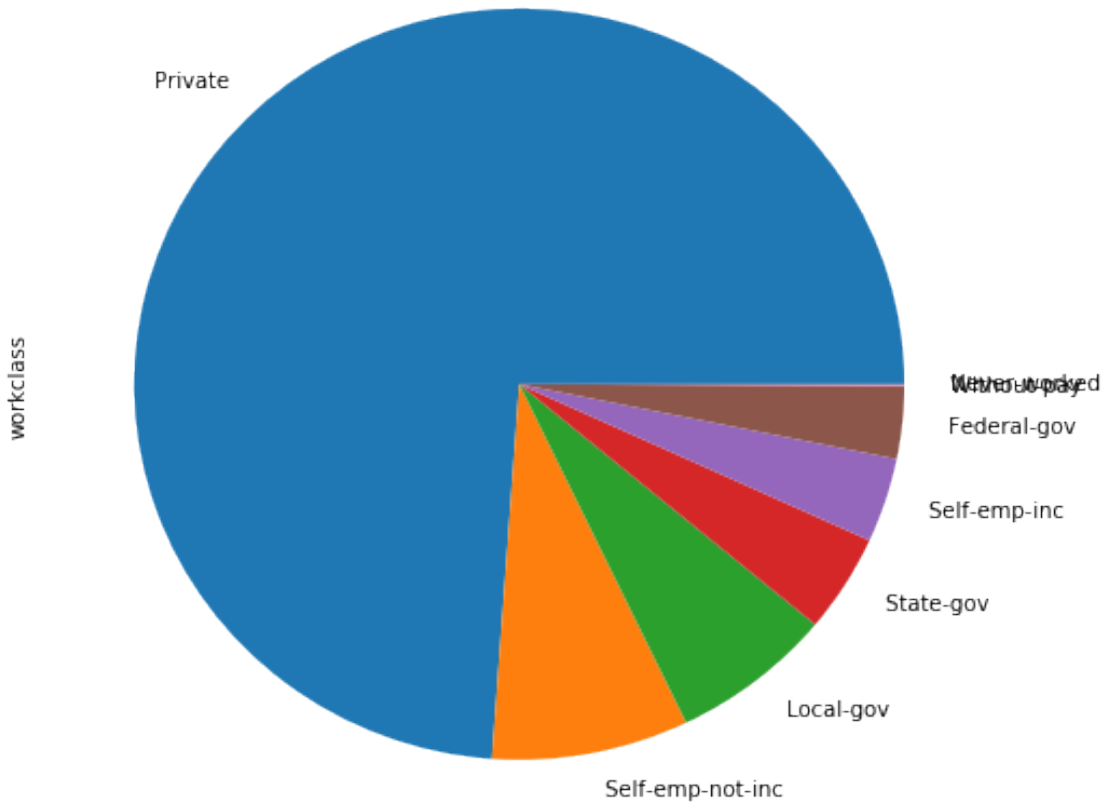
We also need value_counts to plot pie chart

```
In [11]: df["workclass"].value_counts().plot(kind="pie");
```



The chart is too crowded. Let us improve it.

```
In [12]: df["workclass"].value_counts().plot(kind="pie", figsize=(8,8));
```



In [ ]:

In [ ]:

Let us switch to cancer dataset to practice plotting scatter plots and box plots.

In [13]:
```
df = pd.read_csv("cancer_data.csv")
df.head().T
```

Out[13]:

	0	1	2	3	4
id	842302	842517	84300903	84348301	84358402
diagnosis	M	M	M	M	M
radius_mean	17.99	20.57	19.69	11.42	20.29
texture_mean	NaN	17.77	21.25	20.38	14.34
perimeter_mean	122.8	132.9	130	77.58	135.1
area_mean	1001	1326	1203	386.1	1297
smoothness_mean	0.1184	0.08474	0.1096	NaN	0.1003

compactness_mean	0.2776	0.07864	0.1599	0.2839	0.1328
concavity_mean	0.3001	0.0869	0.1974	0.2414	0.198
concave_points_mean	0.1471	0.07017	0.1279	0.1052	0.1043
symmetry_mean	0.2419	0.1812	0.2069	0.2597	0.1809
fractal_dimension_mean	0.07871	0.05667	0.05999	0.09744	0.05883
radius_SE	1.095	0.5435	0.7456	0.4956	0.7572
texture_SE	NaN	0.7339	0.7869	1.156	0.7813
perimeter_SE	8.589	3.398	4.585	3.445	5.438
area_SE	153.4	74.08	94.03	27.23	94.44
smoothness_SE	0.006399	0.005225	0.00615	NaN	0.01149
compactness_SE	0.04904	0.01308	0.04006	0.07458	0.02461
concavity_SE	0.05373	0.0186	0.03832	0.05661	0.05688
concave_points_SE	0.01587	0.0134	0.02058	0.01867	0.01885
symmetry_SE	0.03003	0.01389	0.0225	0.05963	0.01756
fractal_dimension_SE	0.006193	0.003532	0.004571	0.009208	0.005115
radius_max	25.38	24.99	23.57	14.91	22.54
texture_max	NaN	23.41	25.53	26.5	16.67
perimeter_max	184.6	158.8	152.5	98.87	152.2
area_max	2019	1956	1709	567.7	1575
smoothness_max	0.1622	0.1238	0.1444	NaN	0.1374
compactness_max	0.6656	0.1866	0.4245	0.8663	0.205
concavity_max	0.7119	0.2416	0.4504	0.6869	0.4
concave_points_max	0.2654	0.186	0.243	0.2575	0.1625
symmetry_max	0.4601	0.275	0.3613	0.6638	0.2364
fractal_dimension_max	0.1189	0.08902	0.08758	0.173	0.07678

In [14]:
```
df.info()
```

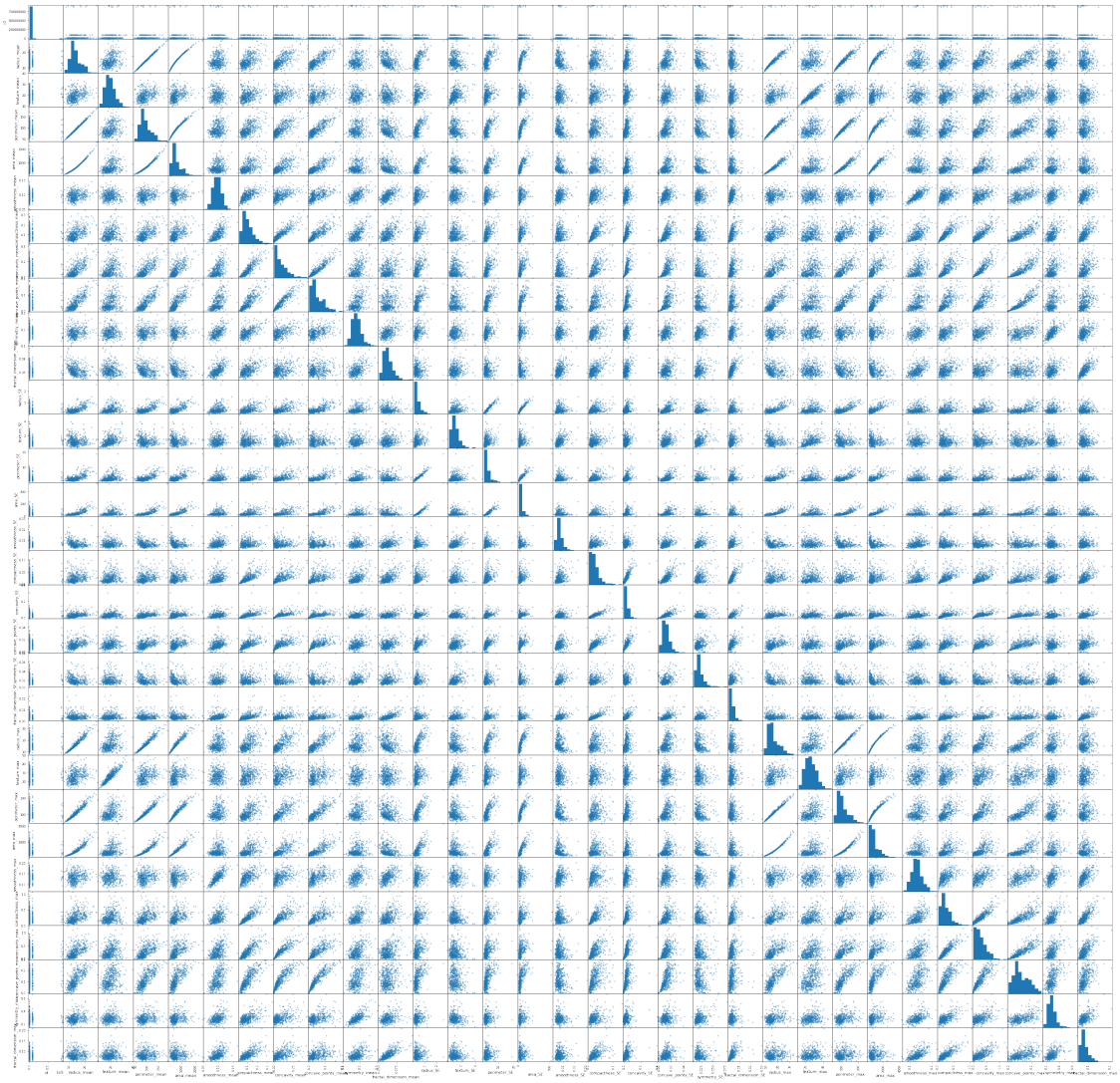
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
id                        569 non-null int64
diagnosis                 569 non-null object
radius_mean               569 non-null float64
texture_mean              548 non-null float64
perimeter_mean            569 non-null float64
area_mean                 569 non-null float64
smoothness_mean           521 non-null float64
compactness_mean          569 non-null float64
concavity_mean            569 non-null float64
concave_points_mean       569 non-null float64
symmetry_mean             504 non-null float64
fractal_dimension_mean    569 non-null float64
radius_SE                 569 non-null float64
texture_SE                548 non-null float64
perimeter_SE              569 non-null float64
area_SE                   569 non-null float64
smoothness_SE             521 non-null float64
```

```
compactness_SE            569 non-null float64
concavity_SE              569 non-null float64
concave_points_SE         569 non-null float64
symmetry_SE               504 non-null float64
fractal_dimension_SE      569 non-null float64
radius_max                569 non-null float64
texture_max               548 non-null float64
perimeter_max             569 non-null float64
area_max                  569 non-null float64
smoothness_max            521 non-null float64
compactness_max           569 non-null float64
concavity_max             569 non-null float64
concave_points_max        569 non-null float64
symmetry_max              504 non-null float64
fractal_dimension_max     569 non-null float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.3+ KB
```

In [ ]:

This next function is really cool for getting quick insight into the relationships amoung numerical variables with scatter plots.

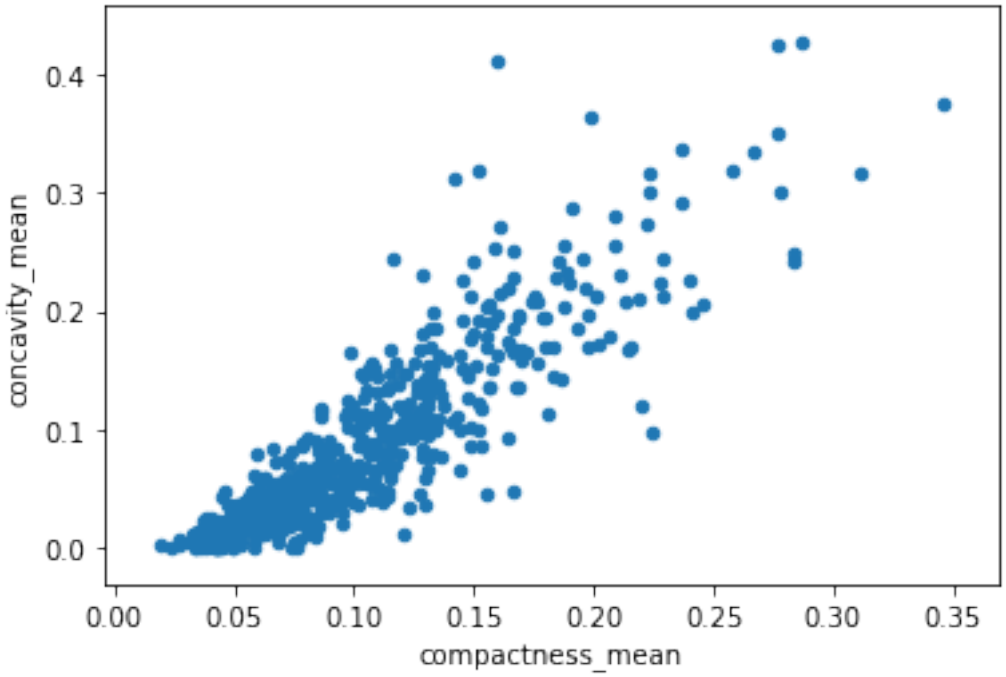
In [15]:
```
pd.plotting.scatter_matrix(df, figsize=(50,50));
```



In [ ]:

We can create a single scatter plot using the standard `.plot()` function with parameters to specifu the columns to be used for the x and y axes

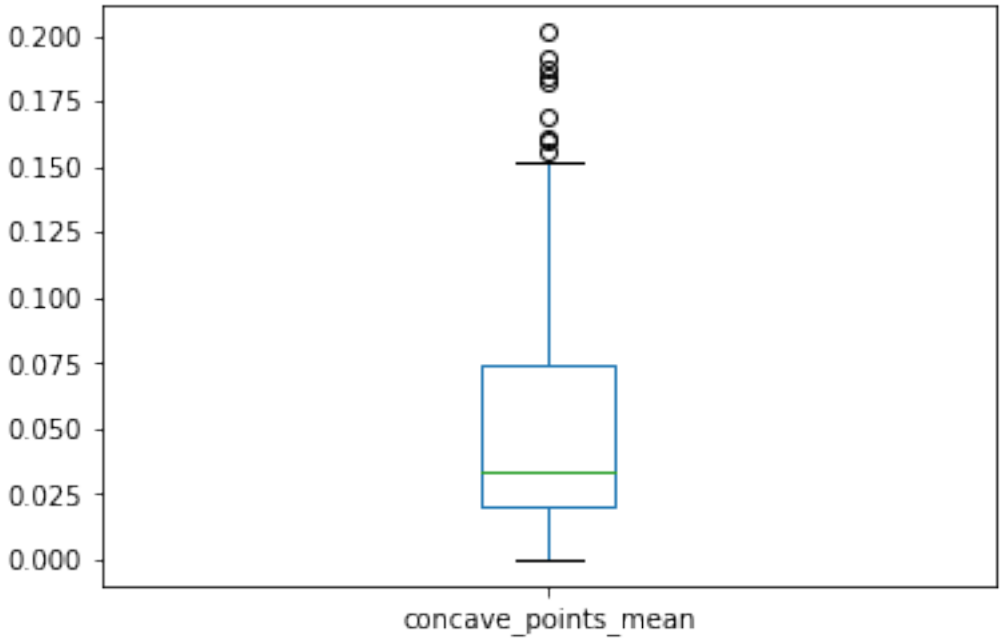
```
In [16]: df.plot(x="compactness_mean", y="concavity_mean", kind="scatter");
```



In [ ]:

The `.plot()` function can also be used to create a box plot.

```
In [17]: df["concave_points_mean"].plot(kind="box");
```



In [ ]:

In [ ]: